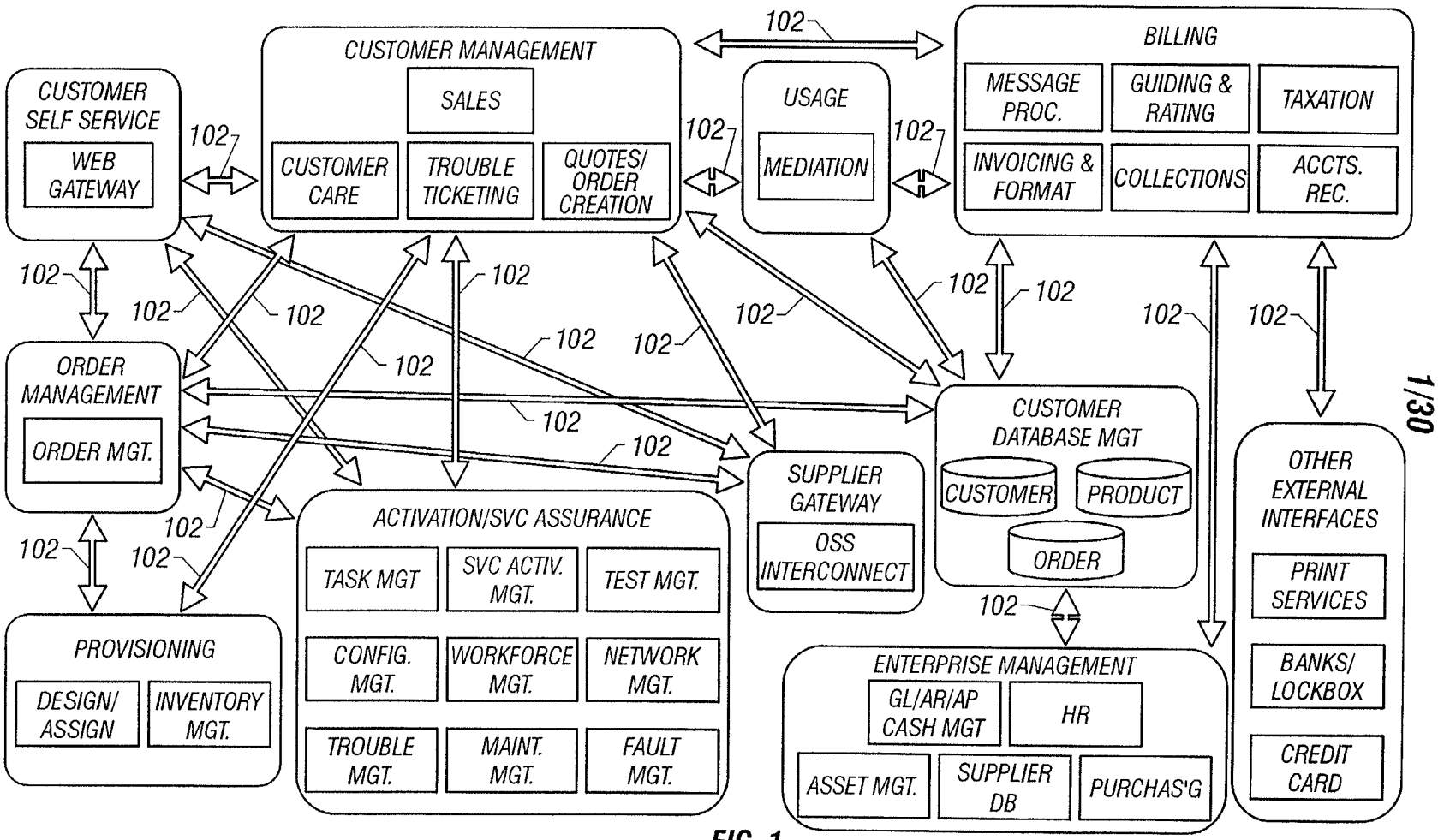

FIG. 1
(Prior Art)

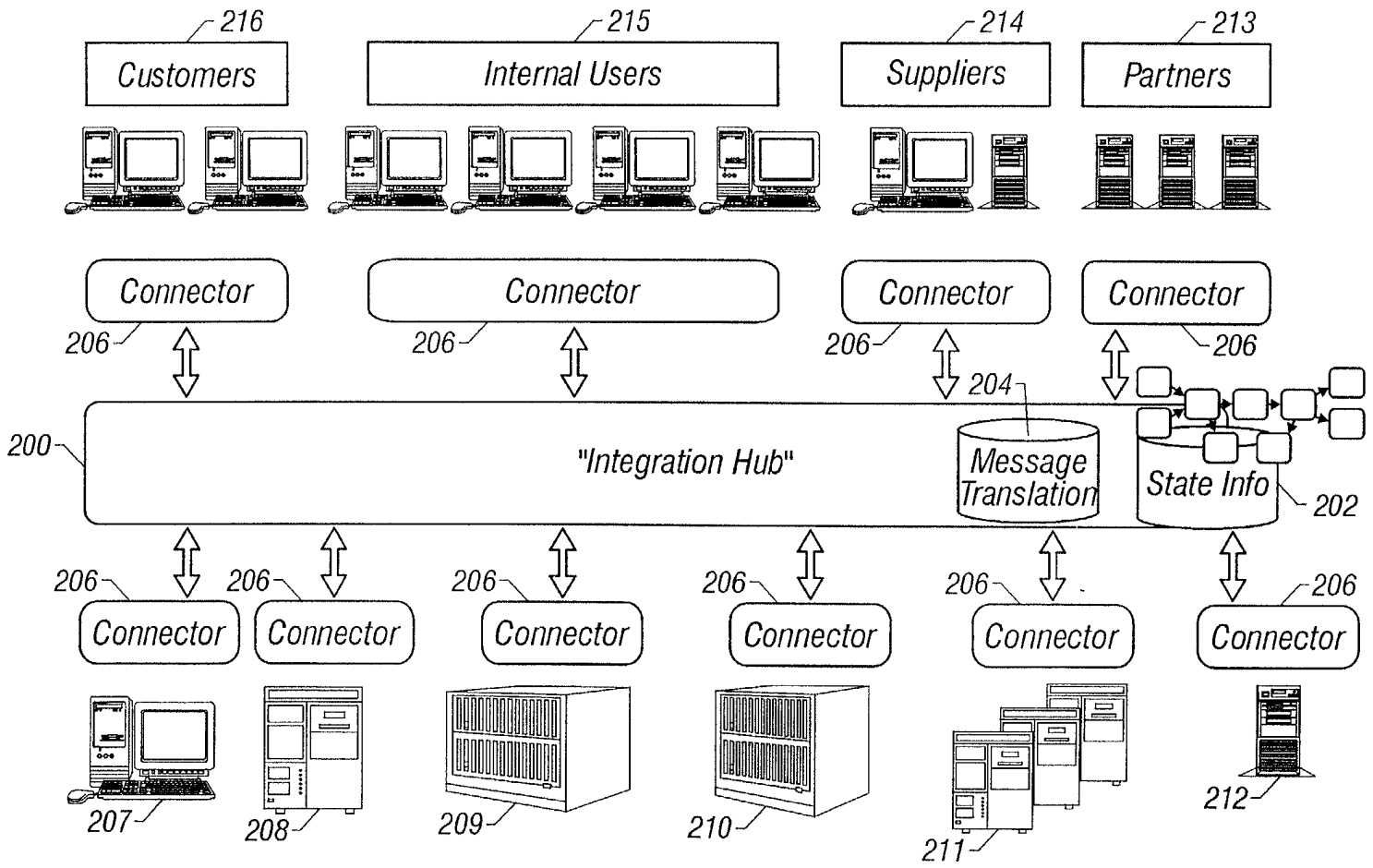
216
Customers
215
Internal Users
214
Suppliers
213
Partners
Connector
206
204
200
"Integration Hub"
Message Translation
State Info
202
207
208
209
210
211
212

FIG. 2

Transformer Class: *SIToVtAddProductTransformer*

Description: *This class directs the data transformation from the Siebel Add Product system event to the Vitria Add Product business event. The Siebel Controller Class will invoke this class when the Add Product system event is received from the Siebel Servlet Source. This class will perform two main functions:*

1. *Invoke the S1IDLBuilder class to transform data from the Siebel Integration Object format to IDL format(Please refer to the IDL Detail Design document for IDL format.)*
2. *Return information in IDL format to the Siebel Controller Class to be published to a channel.*

Class Variables: *None*
Instance Variables: *None*
Method(s): *1. performWork*
Description: *This Method will transform the Vitria Product business component fields retrieved for the Add Product system events to IDL fields to be published by Vitria.*

Method Details
Exception(s): *1.* ***Type:*** *AcknowledgementException*
Description: *This contains the account id, business event, external application, success indicator, element, element id, error message, bpid and eventXML.*

Parameter(s): *1.* ***Name:*** *pEventBody*
Type: *EventBody*
Description: *The eventBody parameter holds the event specifications such as the name of the event, and the parameters of the event.*
2. ***Name:*** *pSti*
Type: *SimpleTranslatorInterface*
Description: *The SimpleTranslatorInterface contains the log level and is used this to write log messages to the appropriate log file.*

Event Body Parameter(s): *1.* ***Name:*** *SiebelMessage*
Type: *SiebelMessage Event*
Description: *The SiebelMessage parameter contains two sub parameters i.e., a sequence of Product Instance struct and attributes of the Siebel Message event.*

Return Value(s): *1.* ***Name:*** *eventBody*
Type: *EventBody*

FIG. 2A

Controller Class: *PIController*

Description: *The Portal Controller (PIController) class will be developed to direct the business events from Vitria to the correct transformer class. Each business event has a different process that needs to be performed to create the appropriate system events. There are two functions that this class performs:*

1. *Directs events to the appropriate transformer class. Each business event will have a separate transformer class that will process the business event into the appropriate system event.*
2. *The system event (i.e. vtOpCode event) is received back from the transformer class. This class will then return the event to the connection model to be sent to the next flow (Portal Target Flow).*

Class Variables: *None*

Instance Variables: *None*

Method(s):

1. getTargetEventInterfaceList

Description: *This method is a public static method which will set the default input Event Interfaces for the Data Transformation flow when the PIController class is selected.*

2. getSourceEventInterfaceList

Description: *This method is a public static method which will set the default output Event Interfaces for the Data Transformation flow when the PIController class is selected.*

3. targetTranslation

Description: *This method will decide what transformer class corresponds to the input event. The transformer name and event are then passed to the callTranslation method. The targetTranslation method will be a public method.*

FIG. 2B-1

4. *callTranslation*
Description: *This method will direct the event to the specified event transformer class and call the applicable transformer method. The callTranslation method is a private method within this class.*
5. *sourceTranslation*
Description: *This method overrides the sourceTranslation method in the super class. We must include this method in the PIController class because it is declared as abstract in the BaseController. However, we will not perform any actual data manipulation at this time, so we will simply return null.*
6. *getEventXML*
Description: *This method will retrieve the source event, and translate it into a xml string.*

Method Details

1. getTargetEventInterfaceList

Exception(s): *None*

Parameter(s):

1. ***Name:*** *strMethodName*
Type: *String*
Description: *The name of the method which was selected by the user within the Processing Tab for a Simple Transformer in the Vitria BW console.*
2. ***Name:*** *translateParams*
Type: *String[]*
Description: *This parameter is not used in this method, but is required by Vitria BW to implement this method.*
7. ***Name:*** *resolver*
Type: *MetaResolver*
Description: *The MetaResolver handles the switch between editable and registered versions of the event list.*

Return Value(s):

1. ***Name:*** *eventBody*
Type: *EventBody*

FIG. 2B-2

2. getSourceEventInterfaceList

Exception(s): *None*

Parameter(s):

1. ***Name:*** *strMethodName*
Type: *String*
Description: *The name of the method which was selected by the user within the Processing Tab for a Simple Transformer in the Vitria BW console.*

2. ***Name:*** *TranslateParams*
Type: *String[]*
Description: *This parameter is not used in this method, but is required by Vitria BW to implement this method.*

3. ***Name:*** *resolver*
Type: *MetaResolver*
Description: *The MetaResolver handles the switch between editable and registered versions of the event list.*

Return Value(s): *None*

3. targetTranslation

Exception(s): *None*

Parameter(s):

1. ***Name:*** *eventBody*
Type: *EventBody*
Description: *The eventBody parameter holds the event specifications such as the name of the event, and the parameters of the event.*

2. ***Name:*** *sti*
Type: *SimpleTranslatorInterface*
Description: *Use this to write log messages to the appropriate log file and retrieve the log level.*

Return Value(s): ***Name:*** *SystemEvent*
Type: *EventBody*

FIG. 2C-1

4. callTranslation

Exception(s): *None*

Parameters(s): *1.* ***Name:*** *eventBody*
Type: *EventBody*
Description: *The eventBody parameter holds the event specifications such as the name of the event, and the parameters of the event.*
2. ***Name:*** *sti*
Type: *SimpleTranslatorInterface*
Description: *SimpleTranslator to retrieve log level and write log messages.*

Return Value(s): ***Name:*** *SystemEvent*
Type: *EventBody*

5. sourceTranslation

Exception(s): *None*

Parameters(s): *1.* ***Name:*** *eventBody*
Type: *EventBody*
Description: *The eventBody parameter holds the event specifications such as the name of the event, and the parameters of the event.*
2. ***Name:*** *sti*
Type: *SimpleTranslatorInterface*
Description: *SimpleTranslator to retrieve logger instance and log level.*

Return Value(s): *None*

FIG. 2C-2

6. getEventXML

Exception(s): *None*

Parameter(s): *1.* ***Name:*** *sti*
Type: *SimpleTranslatorInterface*
Description: *SimpleTranslator to retrieve logger instance and log level.*

Return Value(s): ***Name:*** *xmlEvent*
Type: *String*

FIG. 2D

Acknowledgement Class: *SIAcknowledgementTransformer*

Description: *This class directs and transforms the Siebel Response Message that results from the completion of a Service Status Notification or Update Product Status event to an Acknowledgement business event.*

The Siebel Controller Class will invoke this class when the Siebel Response Message is received from the Siebel Target Driver. This class will perform two main functions:

1. *Transform information from a Siebel Business component format to an IDL format. (Please refer to the IDL Detail Design document for IDL format.)*
2. *Return information in an IDL format to the Siebel Controller Class to be passed to the Acknowledgement Channel.*

Class Variables:	*None*
Instance Variables:	*None*
Method(s):	*1. performWork* ***Description:*** *This method will transform the incoming Siebel Response Message into IDL fields to be published in Vitria.*
Method Details	
Exception:	*1.* ***Type:*** *AcknowledgementException* ***Description:*** *This contains the account id, business event, external application, success indicator, element, element id, bpid, and eventXML.*

FIG. 2E-1

Parameter(s):	*1.* ***Name:*** *pEventBody* ***Type:*** *EventBody* ***Description:*** *The eventBody parameter holds the event specifications such as the name of the event, and the parameters of the event.* *2.* ***Name:*** *pSti* ***Type:*** *SimpleTranslatorInterface* ***Description:*** *Use this to write log message to the appropriate log file.*
Incoming Siebel ResultsEvent Body Parameters:	*1.* ***Name:*** *status* ***Type:*** *String* ***Description:*** *The status field is either Success or Error, indicating whether the event was processed correctly.* *2.* ***Name:*** *error* ***Type:*** *String* ***Description:*** *The error field contains any error message that occurred while processing the Siebel operation.*
Return Values:	*1.* ***Name:*** *eventbody* ***Type:*** *EventBody[]*

FIG. 2E-2

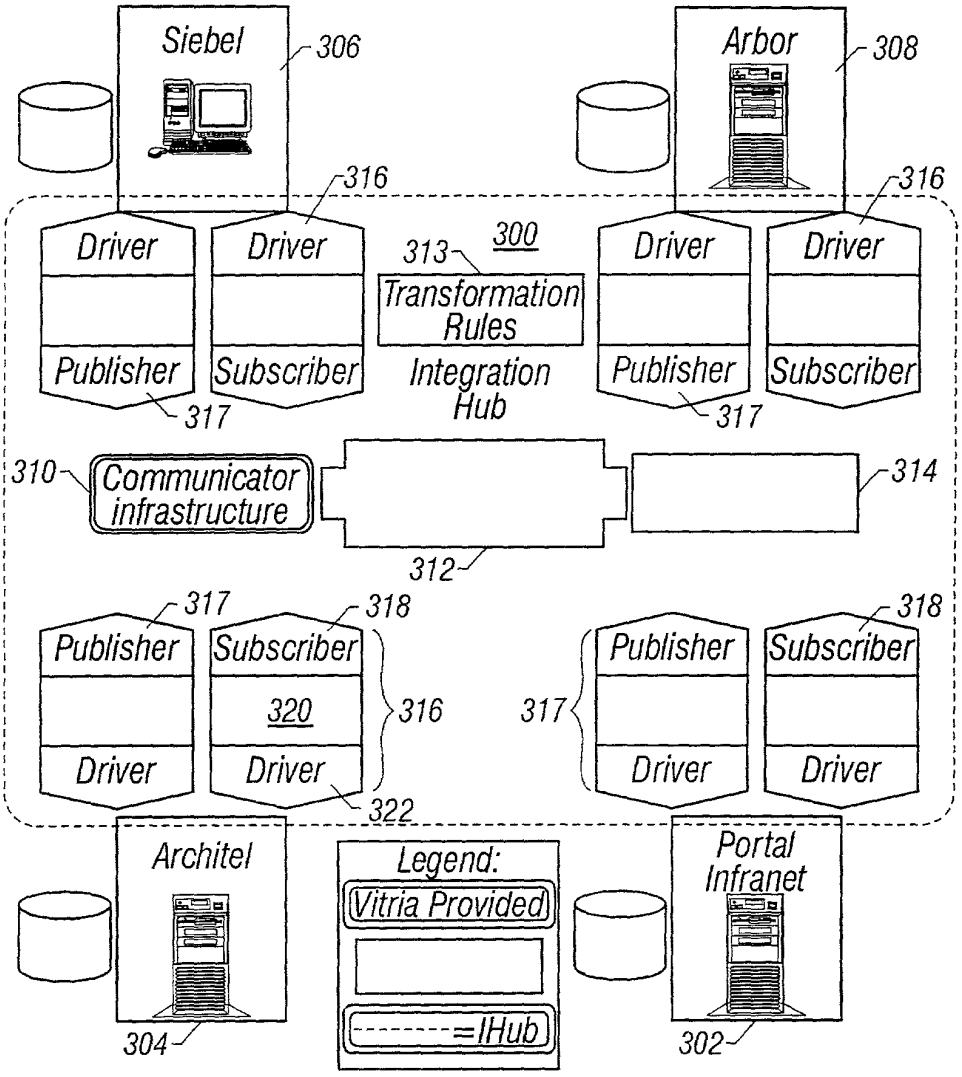
Siebel
306
Arbor
308
316
316
Driver
Driver
313
300
Transformation Rules
Driver
Driver
Publisher
Subscriber
Integration Hub
Publisher
Subscriber
317
317
310
Communicator infrastructure
314
312
317
318
318
Publisher
Subscriber
Publisher
Subscriber
320
316
317
Driver
Driver
Driver
Driver
322
Architel
Legend:
Vitria Provided
Portal Infranet
=IHub
304
302

FIG. 3

account.billinfo.billMedia	R2 Vitria Account		0 PIN_FLD_PAYINFO ARRAY [0] 1 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] 2 PIN_FLD_INV_INFO ARRAY [0] 3 PIN_FLD_DELIVERY_PREFER STR [0]	Vitria must translate the paper type to enumerator
account.parentId	R2 Vitria Account	N/A	0 PIN_FLD_BILLINFO ARRAY [0] 1 PIN_FLD_PARENT POID [0]	This value will be used to retrieve parent id from portal
account.billinfo.currency	R2 Vitria Account	Transform to COM LOV (see IDL doc)	0 PIN_FLD_BILLINFO ARRAY [0] 1 PIN_FLD_CURRENCY INT [0]	String to Integer Transform to COM LOV (see IDL doc)

FIG. 3A-2

account.billInfo.billFrequency	R2 Vitria Account	N/A	0 PIN_FLD_BILLINFO ARRAY [0] 1 PIN_FLD_BILL_WHEN INT [0]	Convert string to ENUM value (see IDL)
account.contactInfo.salutation	R2 Vitria Account	N/A	0 PIN_FLD_NAMEINFO ARRAY [0] 1 PIN_FLD_SALUTATION STR [0]	N/A

FIG. 3B-1

account.companyName	R2 Vitria Account	Limit to 56 char in GUI	0 PIN_FLD_NAMEINFO ARRAY [0] 1 PIN_FLD_COMPANY STR [0]	N/A
account.contactInfo.homePhone	R2 Vitria Account	N/A	0 PIN_FLD_NAMEINFO ARRAY [0] 1 PIN_FLD_PHONES ARRAY [0...N] 2 PIN_FLD_TYPE ENUM [0] 2 PIN_FLD_PHONE STR [0]	In addition, convert field name to type (i.e. home, work, cell)
account.accountType	R2 Vitria Account	'Billing Aggregator' = Unbillable 'Billing' = Billable	0 PIN_FLD_BILLINFO ARRAY [0] 1 PIN_FLD_BILL_TYPE ENUM [0]	Convert string to ENUM value (see IDL)
account.billInfo.ccardExpire Date	R2 Vitria Account	Transform to common date format	0 PIN_FLD_PAYINFO ARRAY [0] 1 PIN_FLD_INHERTIED _INFO SUBSTRUCT [0] 2 PIN_FLD_CC_INFO ARRAY [0] 3 PIN_FLD_DEBIT_EXP STR [0]	Can only accept MM/YY

FIG. 3B-2

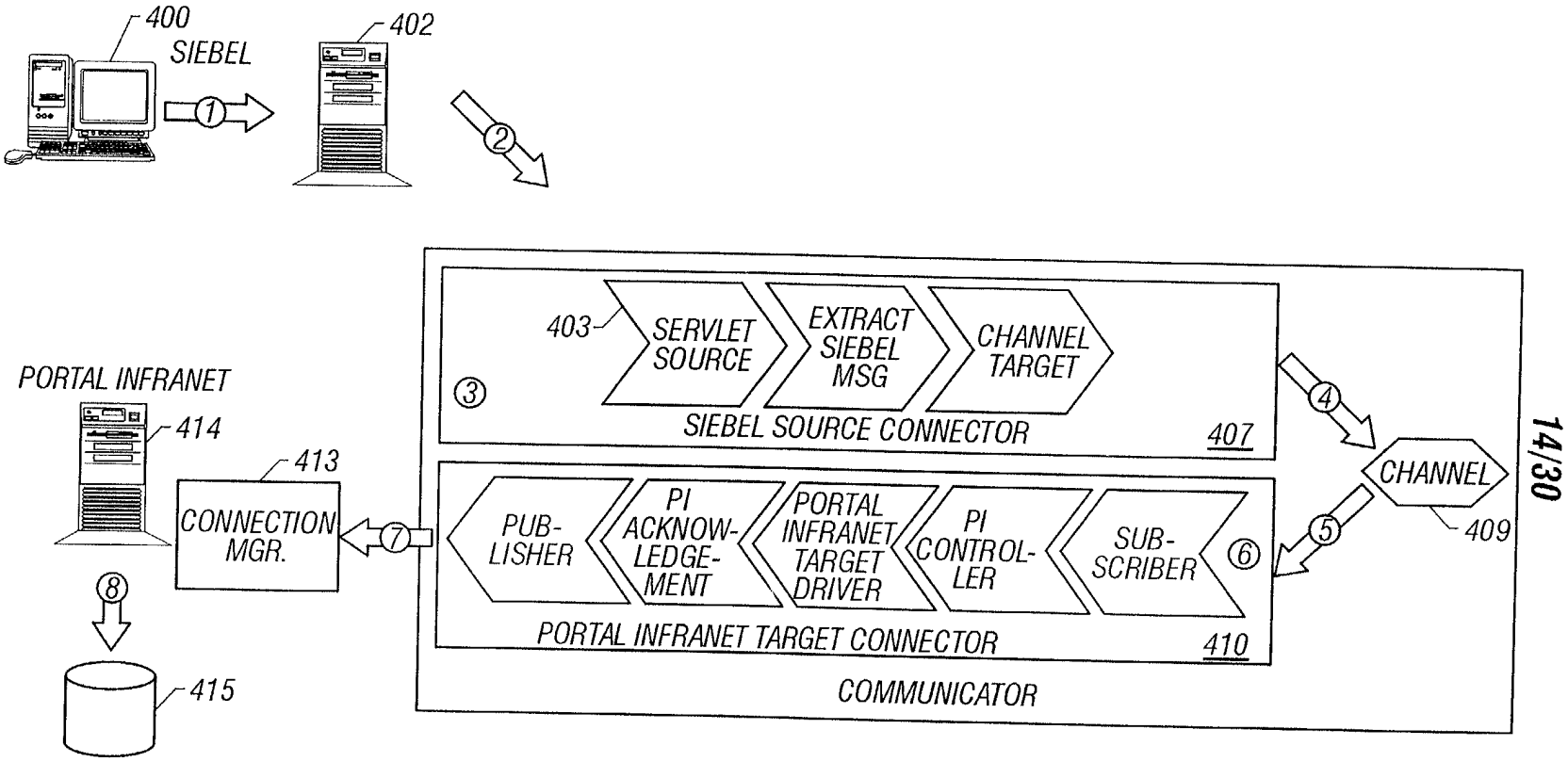
400
SIEBEL
402
1
2
403
SERVLET SOURCE
EXTRACT SIEBEL MSG
CHANNEL TARGET
3
SIEBEL SOURCE CONNECTOR
407
4
CHANNEL
409
PORTAL INFRANET
414
413
CONNECTION MGR.
7
PUB-LISHER
PI ACKNOW-LEDGE-MENT
PORTAL INFRANET TARGET DRIVER
PI CONTROL-LER
SUB-SCRIBER
6
5
PORTAL INFRANET TARGET CONNECTOR
410
8
415
COMMUNICATOR

FIG. 4

Sequence	System Event	Application	Description
1	Acknowledgement	Siebel	After processing a business event, the Siebel target connector will send an Acknowledgement for this event to Vitria to denote whether or not the operation completed successfully.
2a.1	Acknowledgement	Portal	After processing a business event, the Portal connector will send an Acknowledgement for this event to Vitria to denote whether or not the operation completed successfully.
2b.1	Acknowledgement	Arbor	After processing a business event, the Arbor connector will send an Acknowledgement for this event to Vitria to denote whether or not the operation completed successfully.

FIG. 5A

Sequence	System Event	Application	Description
1	Add Product	Siebel	The "Add Product" event will send (service element) information to Vitria. This event is triggered in the CRM application when the "Submit" order button is pressed for an order. A Siebel workflow sends the necessary account information to Vitria XML format via an HTTP protocol.
2a.1	CreateOrder.Request	OMS	Vitria sends a ProvisionRequest to OMS through the OMS target connection model which then translates it into a CreateOrder. Request event.
2a.2	CreateOrder.Response	OMS	Upon provisioning, the OMS source connection model receives a CreateOrder. Response event from OMS which is then translated into a ProvisionResponse.
2a.3	ServiceStatusNotification	OMS	OMS will also send to Vitria a ServiceStatusNotification event indicating the status of the account related to the ProvisionRequest.
3b.1	Find Customer	Portal	As a result of the Add Product system events, Vitria receives an account number. Portal uses the account number and the opcode **PCM_OP_CUST_FIND** to determine the poid of the account that is getting new products added.
3b.2	Find Service	Portal	If CRM does not send Portal service information for this business event, Infranet assumes the deal was intended to be attached at the account level, not at the service level. If CRM sends portal service information, it will be necessary to retrieve the service poid in order to attach the product to that service. Portal uses the Service Type, Service Login, Account ID and the opcode **PCM_OP_SEARCH** to determine the poid of the service. *Note: If CRM fails to send service information for a defined service-level deal, Portal will send back an error to Vitria.
3b.3	Find Deal	Portal	This system event uses the deal name to call the opcode **PCM_OP_CUST_POL_GET_DEALS.** This will retrieve all the deals that are available. Internally, the wrapper opcode will iterate through each of the deals in order to match it to the deal's name.

FIG. 5B-1

Se-quence	System Event	Appli-cation	Description
3b.4	Add Deal	Portal	The final opcode that may be called is **PCM_OP_BILL_PURCHASE_DEAL**. This opcode will take the account poid, deal poid and optionally the service poid. This information will be used to add new deals to the account/service specified.
3b.5	Acknowledgement	Portal	A response will be sent back from the Portal to Vitria indicating if the product has been successfully added or not. Vitria-IOM will transform this Acknowledgement event into an ServiceStatusNotification event to send to Siebel.
4c.1a	Insert Account Component	Arbor	The insert account component event will be called when a new component is added at the account level. The component will be associated at the account level, and attached to the dummy package.
4c.1b	Insert Service Component	Arbor	The insert service component will be called when a new component is added to an account at the service level. The component will be associated to the service instance.
4c.2a	Activate Account Component	Arbor	Once the account component has been inserted into the database, it must be activated individually in the activate APIs before billing can begin. Each component must be activated individually.
4c.2b	Activate Service Component	Arbor	Once the service component has been inserted into the database, it must be activated individually in the activate APIs before billing can begin. Each component must be activated individually.
4c.3	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating that the component has been successfully entered or not. An acknowledgement event will be sent to the acknowledgement channel.

FIG. 5B-2

Sequence	System Event	Application	Description
1	Add Service	Siebel	The "Add Service" event will send service instance information to Vitria. This event will be triggered in the CRM application when the "Submit" order button is pressed for an order. A Siebel workflow sends the necessary account information to Vitria XML format via an HTTP Protocol.
2a.1	Insert Service Instance	Arbor	The service instance event will create a new instance of a service instance object, and will associate the service instance id and a service location to a billing account.
2a.2	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating if the service has been successfully entered or not.
3b.1	Find Customer	Portal	As a part of the Add Service system event, CRM passes the Account ID to Vitria. The wrapper opcode uses the Account ID and the opcode **PCM_OP_CUST_FIND** to determine the account POID.
3b.2	Create Service	Portal	The Create Service Portal system event will occur only after all of the products associated with that service have been provisioned (for I-Hub R3.0, 10 products will be provisioned by OMS, all other products provisioning status will be defaulted) and the associated account has been sent to Portal. Once the account POID (account identifier) has been received by Portal, the wrapper opcode calls **PCM_OP_CUST_CREATE_SERVICE** to create a service instance in the infranet database.
3b.3	Acknowledgement	Portal	A response will be sent back from the Portal to Vitria indicating if the service has been successfully entered or not.
4	N/A	IOM	Vitria IOM will utilize the information within the addService event to create a ProvisioningRequest event to be sent to the Provisioning system.

FIG. 5C

Sequence	System Event	Application	Description
1	Add Account Level Adjustment	Siebel	The "Apply Account Level Adjustment" event sends account adjustment information-credits and debits-from Siebel to Vitria. This event will be triggered when the Account Level Adjustment applet is committed in Siebel. The Siebel Workflow sends the account level adjustments to Vitria in XML format through HTTP protocol.
2a	Find Customer	Portal	Portal uses the account number in the opcode **PCM_OP_CUST_FIND** to determine the account POID.
2b	Apply Adjustment	Portal	The adjustment amount, account POID, and reason description are passed into **PCM_OP_BILL_ACCOUNT_ADJUSTMENT** to apply the adjustment to the account.
2c	Acknowledgement	Portal	A response will be sent back from Portal to Vitria indicating whether the adjustment has been successfully applied or not.
3a	Apply Acct Level Adj	Arbor	This event will be used to apply a credit adjustment to an account's overall charges within Arbor. The account-level adjustment will be made by creating the Adjustment API object and calling the Insert Misc function to apply the credit to the appropriate account.
3b	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating whether the adjustment has been successfully applied or not.

FIG. 5D

Sequence	System Event	Application	Description
1	Cancel Deal	Siebel	The "Cancel Product" will be triggered to send information to Vitria when a deal has been cancelled either by clicking "Cancelled" button or by changing the status field in Siebel. Siebel Workflow sends necessary product information to Vitria in XML format through HTTP protocol.
2a	Find Customer	Portal	As a result of the Cancel Product system event, Vitria receives as account number. Portal uses the account number in the opcode **PCM_OP_CUST_FIND** to determine the account POID.
2b	Read Acct Products (Replacing Find Deal)	Portal	The wrapper opcode uses the partial poid of the account object to call **PCM_OP_CUST_READ_ACCT_PRODUCTS (replacing GET_DEALS)** to determine the account deals and **node location.**
2c	Cancel Product	Portal	Pass the account POID, deal POID, effective cancellation date, and (optional) service POID to **PCM_OP_BILL_CANCEL_DEAL** to cancel the deal at the account or service level. Default date to the correct effective date.
2d	Acknowledgement	Portal	A response will be sent back from Portal to Vitria indicating whether the deal has been successfully cancelled or not.
3a.1	Cease Account Component	Arbor	This event will be used to cancel an instance of a component that is associated at the account level within Arbor. The cease function is used to cancel the unique occurrence of the Product Package Account Component API object.
3a.2	Cease Serv Inst Component	Arbor	This event will be used to cancel an instance of a component that is associated at the service instance level within Arbor. The cease function is used to cancel the unique occurrence of the Product Package Service Instance Component API object.
3b	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating whether the deal has been successfully cancelled or not.
4	N/A	IOM	Vitria IOM will utilize the cancelProduct information to create a provisioningRequest event to ensure the proper provisioning tasks are completed to disconnect the product.

FIG. 5E

Sequence	System Event	Application	Description
1	Cancel Service	Siebel	The "Cancel Service" event will be triggered to send information to Vitria when the "Cancel" button for that service is clicked in Siebel. A Siebel Workflow sends the necessary account information to Vitria in XML format via an HTTP protocol.
2a	CreateOrder.Request	OMS	Vitria sends a CreateOrder.Request (Provisioning Request) event to OMS with the expectation of a CreateOrder.Response (ProvisioningResponse) event followed by a ServiceStatusNotification event.
2b	CreateOrder.Response	OMS	A CreateOrder.Response (ProvisioningResponse) will be sent back from OMS to Vitria - See step 2b above.
3a	Find Customer	Portal	Portal uses the account number in the opcode **PCM_OP_CUST_FIND** to determine the account POID.
3b	Cust Read Acct Products	Portal	**PCM_OP_CUST_READ_ACCT_PRODUCTS** uses the partial poid of the account object to determine the hierarchy of services.
3c	Cancel Service	Portal	Pass the account POID, service POID, effective date, status, and status flags to **PCM_OP_CUST_SET_STATUS** to cancel the service.
3d	Acknowledgement	Portal	A response will be sent back from Portal to Vitria indicating whether the service has been successfully cancelled or not.
4a	Cease Service Instance	Arbor	This event will be used to cancel a service instance within Arbor by calling the cease function on the unique occurrence of the service instance object.
4b	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating whether the service has been successfully cancelled or not.

FIG. 5F

Sequence	System Event	Application	Description
1	Create Account	Siebel	The "Create Account" event will be used to send account information from Siebel to Vitria. The Create Account event is triggered in Siebel when the "Submit" order button is pressed for a new customer. A Siebel workflow sends the necessary account information to Vitria in XML format via an HTTP protocol.
2a.1	Insert Account	Arbor	This event will be used to construct the account within Arbor by creating the acct API object and using the insert function to write the account information to the appropriate Arbor database.
2a.2	Insert Product Package	Arbor	Add a product package to an existing account.
2a.3	Activate Product Package	Arbor	Activate the product for the account.
2a.4	Insert Tax Exemption Info	Arbor	Tax exemption information is set for the account by populating the Tax Exemption API object and calling the insert function to write this information to the appropriate Arbor database. There are four tax exemptions, one each for federal, state, county and city.
2a.5	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating whether the account has been successfully created or not.
2b.1	Find Parent (Find Customer)	Portal	If a parent account number is passed across the Vitria interface, then the wrapper opcode will invoke **PCM_OP_CUST_FIND** in order to establish the poid value of the parent account. This event will also determine whether the parent is a member of a group.
2b.2	Commit Customer	Portal	Once all of the poids have been retrieved based on the information passed from the Vitria interface, the wrapper opcode calls **PCM_OP_CUST_COMMIT_CUSTOMER** in order to load all the new customer information into the infranet database.
2b. 3	Move Member	Portal	If the account is a non-Master account, then the wrapper opcode will call **PCM_OP_BILL_GROUP_MOVE_MEMBER** to associate the child account to a parent account.
2b.4	Set Tax Info	Portal	When present, tax exemption information is passed with the account POID and associated with the account with the **PCM_OP_CUST_SET_TAXINFO** opcode. (Tax exemptions are not in scope for R3.0)
2b.5	Acknowledgement	Portal	A response will be sent back from Portal to Vitria indicating whether the account has been successfully created or not.

FIG. 5G

Se-quence	System Event	Appli-cation	Description
1	Modify Account	Siebel	The "Modify Account" event will be used to send changes to a customer's attributes from Siebel to Vitria. Note: Currency cannot be changed for any account. The Modify Account event is triggered in Siebel when a customer's account information is modified. A Siebel workflow sends the necessary account information to Vitria in XML format via an HTTP protocol.
2a.1	Update Account	Arbor	This event will be used to update contact, billing, address, credit card, and other account related information for an existing account in Arbor.
2a.2	Update Tax Exemption Info	Arbor	This event will be used to update tax exemption information in the appropriate Arbor database. There are four tax exemptions, one each for federal, state, county, and city.
2a.3	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating whether the account has been successfully modified or not.
2b.1	Find Customer	Portal	This event pulls the Account ID from Vitria. Once this information has been received, the opcode **PCM_OP_CUST_FIND** is called to determine the POID of the account.
2b.2	Update Customer	Portal	Once the account POID has been retrieved this event calls the opcode **PCM_OP_CUST_UPDATE_CUSTOMER** in order to make any applicable updates to the customer information into the Infranet database.
2b. 3	Set Tax Info	Portal	When needed, tax exemption modifications are passed with the account POID and associated with the account with the **PCM_OP_CUST_SET_TAXINFO** opcode. (This is out of scope for Ihub R3.0)
2b.4	Acknowledgement	Portal	A response will be sent back from Portal to Vitria indicating whether the account has been successfully modified or not.

FIG. 5H

Sequence	System Event	Application	Description
1	Modify Service	Siebel	The "Modify Service" event will be used to send changes to a customer's service from Siebel to Vitria. The "Modify Service" event will be automatically triggered in Siebel to send information to Vitria when any of the service information is modified. A Siebel Workflow sends the modified service information to Vitria in XML format through HTTP protocol.
2a	Find Customer	Portal	Portal uses the account number in the opcode **PCM_OP_CUST_FIND** to determine the account POID.
2b	Find Service	Portal	Portal will need to retrieve the service poid, using the service information sent from Siebel, in order to attach the product to the appropriate service. Portal uses the opcode **PCM_OP_SEARCH** to determine the poid of the service. The input parameters for **PCM_OP_SEARCH** are the Service Type, the Service Login, an inline search template built into the opcode's input flist. *Note: In the case that Siebel fails to send service information for a defined service-level deal, Portal will send back an error to Vitria.
2c	Modify Service	Portal	Set the service type for the service object in the **PCM_OP_CUST_MODIFY_SERVICE** opcode by passing it to the service POID and the INHERITED_INFO array. The INHERIT_INFO array contains the effective date and service location information.
2d	Set Password	Portal	Set the service password in **PCM_OP_CUST_SET_PASSWD** opcode by passing it the account POID, service POID, and password.
2e	Acknowledgement	Portal	A response will be sent back from Portal to Vitria indicating whether the service has been successfully modified or not.
3a	Modify Service Instance	Arbor	This event will be used to modify information related to the service instance within Arbor.
3b	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating whether the service has been successfully modified or not.

FIG. 5I

Se-quence	System Event	Appli-cation	Description
1	Service Status Notification	OMS	The OMS Source Connector will monitor the OMS database to see if there is any change in the status of a service or product. If there is a change in the status of a service or product, a Service Status Notification Event is created with the required parameters. The OMS server is accessed to obtain additional information. The Service Status Notification Event is then sent to the Vitria IOM for handoff to the automator.
2	Service Status Notification	Vitria IOM	Vitria IOM will be responsible for creating and sending the Update Product Status Business Event to Siebel (for CRM purposes). Vitria IOM logic will determine the status response to be sent upstream to Siebel. When the status is "Active", the Vitria IOM will inform Portal/Arbor to begin the billing process.
3	Update Product Status	Siebel	The Update Product Status Event will send product update information from Vitria to Siebel. Vitria will update the CRM application to "Pending" once OMS has received a work order. Vitria will also update the CRM to "Active" once the order has been successfully provisioned in OMS. Vitria will send this information to Siebel in XML format via an HTTP protocol.

FIG. 5J

Sequence	System Event	Application	Description
1	Update Account Status	Siebel	The Update Account Status event sends account status information from Siebel to Vitria. The Update Account Status event is triggered automatically in Siebel when a customer's account status is changed. A Siebel workflow sends the necessary account information to Vitria in XML format via an HTTP protocol.
2	N/A	IOM	The Account Process Model will receive the updateAccountStatus event and send it off to billing. It will wait for a successful acknowledgement event in return from the billing applications to be notified with the changes are complete in billing.
3a	Find Customer	Portal	Portal uses the account number in the opcode **PCM_OP_CUST_FIND** to determine the account POID.
3b	Update Account Status	Portal	Pass the account POID, status, status flag, and defaulted program name to **PCM_OP_CUST_SET_STATUS** to set the account status to active, suspended, or closed.
3c	Acknowledgement	Portal	A response will be sent back from Portal to Vitria indicating whether the account status has been successfully changed or not.
4a	Cease Account	Arbor	This event will be used to update the status of an account to a disconnect requested status by utilizing the cease function on the Account API object.
4b	Reactivate Account	Arbor	This event will be used to update the status of an account to a current or active status from a closed status by utilizing the reactivate function on the Account API object.
4c	Acknowledgement	Arbor	A response will be sent back from Arbor to Vitria indicating whether the account status has been successfully changed or not.

FIG. 5K

Se-quence	System Event	Appli-cation	Description
1	Update Product Status	Siebel	Siebel will send an event to Vitria IOM that will trigger the update product event. Once Vitria receives this event it will send it on to the provisioning system. Vitria IOM will then wait until it receives a provisioning request response from the provisioning system (OMS), indicating that the event was received. It will then send a message onto Siebel letting it know that the request has been received. It will then wait for a serviceStatusNotification response from the provisioning system. Once it has received the response it will send the update product Status to Siebel to change the status to either complete or canceled.

FIG. 5L

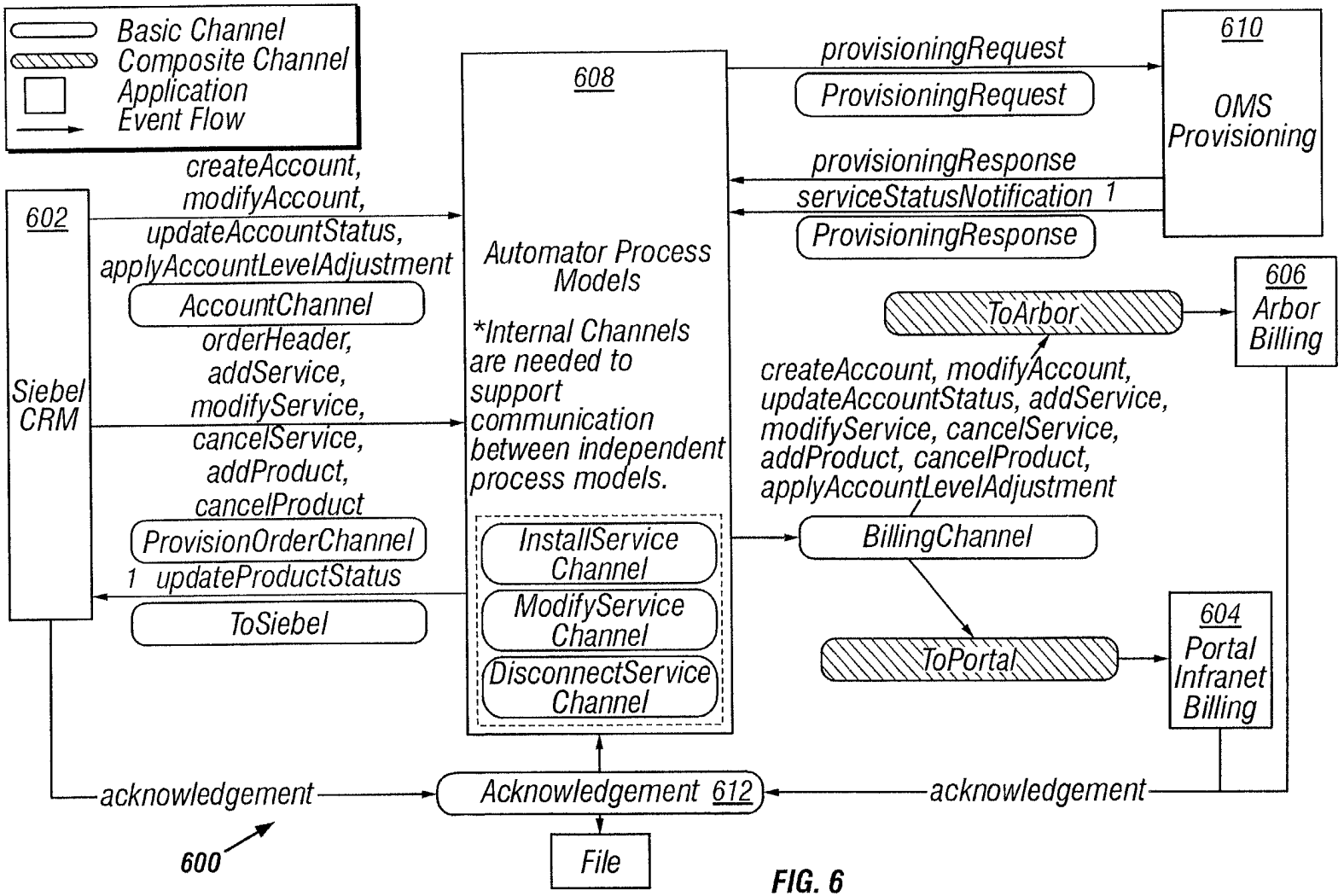
Basic Channel
Composite Channel
Application
Event Flow
602
Siebel CRM
createAccount,
modifyAccount,
updateAccountStatus,
applyAccountLevelAdjustment
AccountChannel
orderHeader,
addService,
modifyService,
cancelService,
addProduct,
cancelProduct
ProvisionOrderChannel
1 updateProductStatus
ToSiebel
608
Automator Process Models
*Internal Channels are needed to support communication between independent process models.
InstallService Channel
ModifyService Channel
DisconnectService Channel
provisioningRequest
ProvisioningRequest
provisioningResponse
serviceStatusNotification 1
ProvisioningResponse
610
OMS Provisioning
ToArbor
606
Arbor Billing
createAccount, modifyAccount, updateAccountStatus, addService, modifyService, cancelService, addProduct, cancelProduct, applyAccountLevelAdjustment
BillingChannel
ToPortal
604
Portal Infranet Billing
acknowledgement
Acknowledgement 612
acknowledgement
File
600

FIG. 6

Channel Name	Description	Business Event	Publisher	Subscriber
AccountChannel	Events sent to this channel will pertain to creating, modifying, and disconnecting accounts	CreateAccount ModifyAccount UpdateAccountStatus ApplyAccountLevelAdjustment	Siebel Source Connector	Account Process Model
Provision Order Channel	Events sent to this channel will pertain to installing and disconnecting products and services	OrderHeader AddService ModifyService CancelServices AddProduct CancelProduct	Siebel Source Connector	Order Processor Model
ToSiebel	All events that need to be published to the Siebel System	ServiceStatusNotification UpdateProductStatus	Service Processor Model	Siebel Target Connector
Provisioning Request	All events that need to be published to the Provisioning systems	ProvisioningRequest	Service Processor Model	OMS Target Connector
Provisioning Response	All events that are published by Provisioning systems	ProvisioningResponse ServiceStatusNotification	OMSSource Connector OMSTarget Connector	Service Processor Model
Acknowledgement Channel	All Acknowledgements that an event was a success or a failure	Acknowledgement	ArborTarget Connector PortalTarget Connector SiebelTarget Connector	Acknowledgement ToFile Service Processor Model

FIG. 7A

Channel Name	Description	Business Event	Publisher	Subscriber
Billing Channel	All events that need to be published to the Billing systems	CreateAccount ModifyAccount UpdateAccountStatus ApplyAccountLevelAdjustment AddService ModifyService CancelService AddProduct CancelProduct	Service Processor Model	ToArbor Channel ToPortal Channel
ToArbor	All events that need to be published to the Arbor System	CreateAccount ModifyAccount UpdateAccountStatus ApplyAccountLevelAdjustment AddService ModifyService CancelService AddProduct CancelProduct	Billing Channel	Arbor Target Connector
ToPortal	All events that need to be published to the Portal System	CreateAccount ModifyAccount UpdateAccountStatus ApplyAccountLevelAdjustment AddService ModifyService CancelService AddProduct CancelProduct	Billing Channel	Portal Target Connector

FIG. 7B